METODI del GRADIENTE

Pb: $Ax = b$ con $A \in \mathbb{R}^{n\times n}$ simmetrica def. positiva.

So che $\exists!$ la soluzione : $x^* = A^{-1}b$

I metodi del gradiente trasformano il problema nella ricerca di un minimo di un funzionale quadratico.

Si può dimostrare che il nostro problema è equivalente a:

Trovare $\min_x \Phi(x)$ con $\Phi(x) = \frac{1}{2}x^TAx - \underbrace{x^Tb}_{=b^Tx}$ funzionale quadratico

Verifichiamolo:

$$\Phi(\tilde{x}) = \Phi(x+e) = \frac{1}{2}(x+e)^TA(x+e) - b^T(x+e) =$$

$$= \frac{1}{2}(x^TAx + x^TAe + \underbrace{e^TAx}_{=x^TAe \text{ perché } A \text{ è simmetrica}} + e^TAe) - b^Tx - b^Te =$$

$$= \frac{1}{2}x^TAx - b^Tx + \underbrace{x^TAe - b^Te}_{(*)} + \frac{1}{2}e^TAe$$

Se x è soluzione del Pb ho $Ax = b$ e quindi anche $x^TA = b^T$

$$\rightsquigarrow (*) = (x^TA - b^T)e = 0$$

$$= \frac{1}{2}x^TAx - b^Tx + \frac{1}{2}e^TAe = \Phi(x) + \underbrace{\frac{1}{2}e^TAe}_{>0 \text{ perché } A \text{ è def pos}} > \Phi(x)$$

Quindi abbiamo mostrato che se $x^* = A^{-1}b$ allora $\Phi(x^*+e) > \Phi(x^*)$ e quindi $\Phi(x^*)$ è il minimo.

Vediamo lo schema dei metodi iterativi di minimizzazione:

$(*)$ $x_{k+1} = x_k + \alpha_k p_k$ dove p_k è la direzione di ricerca

$\alpha_k \in \mathbb{R}$ opportuno.

def: Dico che p è una **DIREZIONE di DECRESCITA** per $\Phi(x)$ in x se $\exists \alpha_0 > 0$ t.c. $\Phi(x+\alpha p) < \Phi(x)$ $\forall\, 0 < \alpha < \alpha_0$.

PROP: Sia $\varphi \in C^1$. Se $(\nabla\Phi(x))^Tp < 0 \Rightarrow p$ è direzione di decrescita.

dim: $\Phi(x+\alpha p) - \Phi(x) \underset{\text{taylor}}{=} \alpha(\nabla\Phi(x))^Tp + o(\alpha) < 0$ per α opportuno ∎

Ritorniamo a $(*)$, dobbiamo trovare un modo per scegliere α_k e p_k. (Tipicamente) α_k è sempre scelto nello stesso modo, i metodi si differenziano per la scelta di p_k.

SCELTA di α_k: (comune per tutti i metodi, si suppone p_k scelto)

Dato p_k si sceglie α_k t.c. $\Phi(x_{k+1}) = \min_{\alpha\in\mathbb{R}} \Phi(x_k + \alpha p_k)$.

È facile calcolare α_k per i funzionali quadratici (altrimenti il problema non sarebbe banale!)

$$\Phi(x_k + \alpha p_k) = \frac{1}{2}(x_k + \alpha p_k)^TA(x_k + \alpha p_k) - b^T(x_k + \alpha p_k)$$

$$\frac{\partial\Phi}{\partial\alpha} = (x_k + \alpha p_k)^TAp_k - b^Tp_k \overset{\text{lo pongo}}{=} 0$$

Trovo un estremante, che è un minimo perché $\Phi(x_k + \alpha p_k)$ è una parabola in α a coefficiente $p_k^TAp_k > 0$.

Quindi ottengo: $\alpha_k = \frac{b^T p_k - x_k^T A p_k}{p_k^T A p_k} \overset{A=A^T}{=} \frac{(b-Ax_k)^T p_k}{p_k^T A p_k}$

Ricordando cos'è il RESIDUO, $r(x) = b - Ax$ e ponendo $r_k = r(x_k) = b - Ax_k$ possiamo riscrivere α_k in una forma più compatta:

$$\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k}$$

OSS: Se p_k è una direzione di decrescita allora

$$r_k^T p_k = -(\nabla \Phi(x_k))^T p_k \overset{\text{def dir. di decr.}}{>} 0$$

e quindi $\alpha_k > 0$ (perché il den è sempre >0 perché A def. pos.)
(Quindi la scelta di α_k che sto facendo va bene.)

METODO STEEPEST DESCENT (più ripida discesa)

Si sceglie $p_k = -\nabla \Phi(x_k) = r_k$ massima pendenza.

N.B: Dai problemi grandi A è mal condizionata, quindi in realtà non ci va bene (diciamo che ha solo importanza "storica" ma non lo usa nessuno).
In pratica se A è ben condizionata qualsiasi metodo va bene, il punto è trovarne uno che funziona anche se A non è ben condizionata!

Quello che in pratica succede è che se il problema è mal condizionato può capitare che continui a ripercorrere le stesse direzioni, senza prendere di nuovo.
Questo perché ho sempre che $p_{k-1} \perp p_k$ e quindi può succedere che $p_{k+1} // p_{k-1}$ (memo: sono in dim n, e quindi non è detto che succeda per forza)

L'idea del gradiente coniugato, che lo rende meglio, è quella di correggere p_k in modo che tenga conto delle direzioni precedenti per non ripercorrerle

METODO GRADIENTE CONIUGATO

Il problema risiede la scelta di p_k.

def: Data $A \in \mathbb{R}^{n\times n}$ sdp, dico che $p, q \in \mathbb{R}^n$ sono **A-ORTOGONALI** se

$$p^T A q = 0.$$

Questa definizione, è una definizione di ortogonalità che tiene conto del problema. È come se misurassi attraverso il problema.

Poniamo $p_k = r_k + \beta_k p_{k-1}$

in modo da tener conto della direzione precedente.
Più precisamente:

$$\begin{cases} p_0 = r_0 \\ p_k = r_k + \beta_k p_{k-1} \end{cases}$$

con β_k t.c. $p_k^T A p_{k-1} = 0$.

Questo perché vogliamo che le direzioni siano A-ortogonali. ($p_k \perp_A p_{k-1}$)

Per prima cosa vediamo esplicitamente com'è β_k.

$$0 = p_k^T A p_{k-1} = (r_k + \beta_k p_{k-1})^T A p_{k-1} \Rightarrow \beta_k = -\frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$$

Ora devo verificare che effettivamente p_k è una direzione di decrescita:

$$(\nabla\phi(x))^T p_k = -r_k^T(r_k+\beta_k p_{k-1}) =$$
$$\stackrel{!}{=} -r_k^T r_k - \beta_k r_k^T p_{k-1} =$$

(viene perché abbiamo scelto α_k, non dipende nemmeno da p_k)

Ora dimostreremo che β_k è uguale a zero e quindi

$$\vdots$$
$$= -\|r_k\|_2^2 < 0 \quad \text{ok}$$

PROP: 1) $\forall k \quad r_{k+1}^T p_k = 0$ (vale in generale)

2) $\forall k \quad r_{k+1}^T r_k = 0$ quindi a coppie i residui sono ortogonali (in senso euclideo) (vale solo per CG)

OSS: $r_{k+1} := b - Ax_{k+1} = b - A(x_k+\alpha_k p_k) = (b - Ax_k) - \alpha_k A p_k =$

$$\stackrel{!}{=} r_k - \alpha_k A p_k$$

dim: (1) $r_{k+1}^T p_k = (r_k - \alpha_k A p_k)^T p_k = r_k^T p_k - \alpha_k p_k^T A p_k = 0$ (PROP)

dove $\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k}$ per def di α_k

(2) $r_{k+1}^T r_k = r_{k+1}^T(p_k - \beta_k p_{k-1}) = r_{k+1}^T p_k - \beta_k r_{k+1}^T p_{k-1} =$ ($r_{k+1}^T p_k = 0$ per (1))

$$\stackrel{!}{=} -\beta_k (r_k - \alpha_k A p_k)^T p_{k-1} = -\beta_k r_k^T p_{k-1} + \alpha_k \beta_k p_k^T A p_{k-1} = 0$$

perché $p_k \perp_A p_{k-1}$ per def dei p_k. ■

Ora vogliamo vedere l'algoritmo del CG. Prima di scrivere l'algoritmo vediamo delle RISCRITTURE di α_k e β_k per RIDURRE IL COSTO COMPUTAZIONALE. Abbiamo

$$\alpha_k := \frac{r_k^T p_k}{p_k A p_k} = \frac{r_k^T r_k}{p_k A p_k}$$

NB non è che $p_k = r_k$ ma quando faccio il prodotto con r_k^T vengono uguali

$$\beta_k := \frac{-r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$$

← sono spostati i prodotti matriciali che hanno costo computazionale maggiore. In più $r_{k-1}^T r_{k-1}$ l'ho già calcolato al passo precedente.

dim: • $r_k^T p_k = r_k^T(r_k + \beta_k p_{k-1}) = r_k^T r_k + \beta_k r_k^T p_{k-1} = r_k^T r_k$ ($r_k^T p_{k-1} = 0$ per (1)) ↰ vale in generale, perché (1) vale in generale, mi piace perché sto riscrivendo α_k, che non "dipende" dal metodo CG

• $p_k^T r_{k-1} \stackrel{(oss)}{=} p_k^T(r_k + \alpha_{k-1} A p_{k-1}) = p_k^T r_k + \alpha_{k-1} p_k^T A p_{k-1} =$ ($p_k^T A p_{k-1} = 0$)

$$\stackrel{!}{=} p_k^T r_k = r_k^T r_k$$

Ma ho anche:

$$p_k^T r_{k-1} = (r_k + \beta_k p_{k-1})^T r_{k-1} = r_k^T r_{k-1} + \beta_k p_{k-1}^T r_{k-1} = \beta_k r_{k-1}^T r_{k-1}$$ ($r_k^T r_{k-1} = 0$ per (2)) per •

Uguagliando le due quantità si ricava β_k. ■

ALGORITMO CG: pcg di Matlab, lavora in due modi:

- $A, b \rightsquigarrow x$ (eventi M precond)
- A, b (M) + function per fare i 2 pt.

• Calcola Ax con prodotto mat.-vett.
• $Mz = r$ con LU